CHAPTER 13
ERROR DETECTION

In all electrical information transmission systems, due consideration must be given to the effects of noise. Noise is any unwanted signal and may originate from sources as spectacular as lightning strikes or as mundane as dirty contacts on telephone switching equipment.

The most important characteristic of noise in telecommunications systems is the relatively long duration of the disturbances. A noise burst of .01 second duration is not uncommon and sounds like a simple click during a voice conversation. If the .01 second noise burst occurs during a 4800 bit per second data transmission, the "simple click" is the death knell for 48 data bits. Thus, when noise causes bits to be received in error, it generally causes a great number of bits to be affected. The periods of high error rate are generally separated by relatively long intervals of low noise, low error rate data reception. Thus the error rate averaged over an hour is typically one error bit in 100,000 bits received.

The "bursty" nature of errors in telecommunications systems is very important in considering error detection.

To determine whether the bits in a character have been properly received, it would be quite simple to append an additional bit to each character and to have that bit be a one or a zero according to the rule that "all transmitted characters shall have an odd number of ones." Thus, for example, the character 01001100 would be expanded to $\underline{0}01001100$ and the character 01101100 would be expanded to

1̲01101100. The added, underlined bit is called the "parity" bit and using it to make the number of ones odd is called "odd parity." (Plainly, one could make the number of ones even and call it "even parity.") In a parity system, the transmitter unit calculates the state of the parity bit and appends it to the character during transmission. The receiving unit calculates the state of the parity bit and compares the calculated value to the value actually received. If they disagree, the receiver knows that a bit has been received in error.

Let us assume that the transmitter sends 101101100. Parity is odd; everything is OK. Let us assume that the second and third bits (counting from the right) are received erroneously. The received character is then 101101010. The parity is still odd and things appear OK despite the double error. The lesson to be learned is that parity on each character can detect only errors that affect a single bit (or three, or five, etc.). Errors that affect two (or four, or six, etc.) bits will not be detected. This problem exists regardless of whether "odd parity" or "even parity" is used.

Consider transmission of the six characters in Figure 13-1. Each bit (except the left-most) in the Check Character has been computed such that it and the bits immediately above it in Characters 1 through 5

101101100	Character 1
110101111	Character 2
001110101	Character 3
111100010	Character 4
100010111	Character 5
010111100	Check Character

Figure 13-1. Sample Transmission

total an odd number of ones. For example, the right-most bit in the Check Character is a zero because three of the right-most bits in the characters above it are ones; hence there is an odd number of ones.

In all characters, including the Check Character, the left-most bit is a parity bit, as described above.

Let us assume that the transmitter sends 101101100 (Character 1). Parity is odd; everything is OK. Let us assume that the second and third bits (counting from the right) are received erroneously. The received character is then 101101010. The parity is still odd and things appear OK despite the double error. This time, however, there is a Check Character being sent. The error just described increases the number of ones in the second column from three to four (an even number) and decreases the number of ones in the third column (counting from the right) from four to three (excluding the check character).

When the Check Character sent by the transmitter is compared in the receiver to that which the receiver has calculated from summing the ones in the various "columns" of the received characters, the second and third bit positions will be incorrect. The computed Check Character will require the second bit from the right to be a "1" to increase the number of ones in that column from four to five, and will require the third bit to be a "0" as the receiver has received only three one bits in that column. Thus the calculated Check Character and the transmitted Check Character will disagree in the second and third bit positions, which are indeed where the errors occurred.

Lest one embrace this scheme as foolproof, consider the case where Character 1 and Character 3 are received with errors in the second and third bit positions. Now there will be two more ones in the second column and two fewer ones in the third column. The number of ones in each column is now such that the receiver's calculated block Check Character and that sent by the transmitter will be the same.

Thus in the same way that a parity check within the character was defeated by a double error in the character, parity on the columns (referred to as a Longitudinal Redundancy Check or LRC) is defeated if a double error occurs in a column. There are numerous possibilities for double bit errors in characters to occur simultaneously with double bit errors in columns in such a fashion that neither the character parity (referred to as Vertical Redundancy Check or VRC) nor the column parity (LRC) will indicate that the errors have occurred. This is an especially important problem since errors in communications transmission systems tend to occur in bursts, as noted above.

The detection systems most effective at detecting errors in communications systems with a minimal amount of hardware (but more than VRC/LRC systems) are the Cyclic Redundancy Checks (CRC).

CRC calculations are customarily done in a multi-section shift register which feeds into an exclusive-OR gate whose output feeds back to other exclusive-OR gates located in between the sections of the shift register. Figure 13-2 shows a typical arrangement.

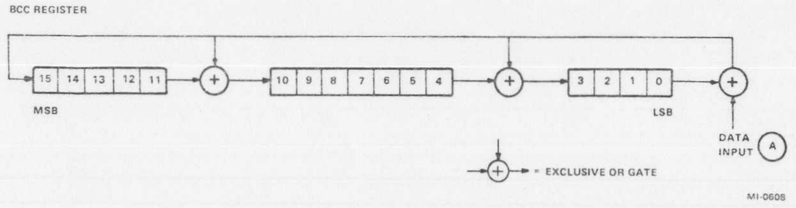


Figure 13-2. Block Check Register Implemented with Shift Registers and Exclusive-OR Gates

The placement and quantity of the exclusive-OR gates vary for CRC-12, CRC-16, and CRC-CCITT, which are the most common Cyclic Redundancy Checks. An exclusive-OR gate is a gate where the output is a 0 if the inputs are both 0 or are both 1. If the inputs differ, the output of the exclusive-OR gate is a 1.

When the logic shown in Figure 13-2 is used in a transmitter circuit, it is initialized to all zeros. As each bit is presented to the communications line it is also applied to the point marked "A" in Figure 13-2 and a shifting pulse is applied to the shift register. Figure 13-3 shows the contents of the shift register assuming that the first bit transmitted was a 1. Note what a dramatic effect just this single bit has and how that effect is spread throughout the register.

As the "1" bits shown in Figure 13-3 get shifted on through the segments of the shift register during the transmission of subsequent bits, they will eventually reach some of the exclusive-OR gates between the segments of the shift register. Here they will affect the state of the "feedback bits" coming from the exclusive-OR gate on the far right-hand side of the diagram.

The general effect to be recognized is that the effect of any
reflected in the various bits of the shift register for a considerabl
after that bit is transmitted.

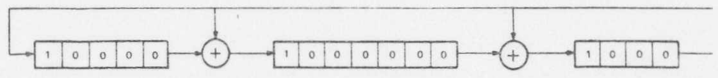


Figure 13-3. Propagation of a "1" Bit into Block Check Regis

In a CRC equipped system, a logical arrangement identical t
used in the transmitter (Figure 13-3) is also used in the receiver.
the register is initialized to zero and data *from* the communic
line is applied to point A while the shift register contents are
once for each bit received. At the conclusion of the message, th
mitting station sends the contents of the transmitter CRC shift r
to the receiving station. The receiving station applies the incom
stream to point A of its CRC logic, just as it did with the pre
data bits. The properties of exclusive-OR gates shown in Figu
are reviewed below.

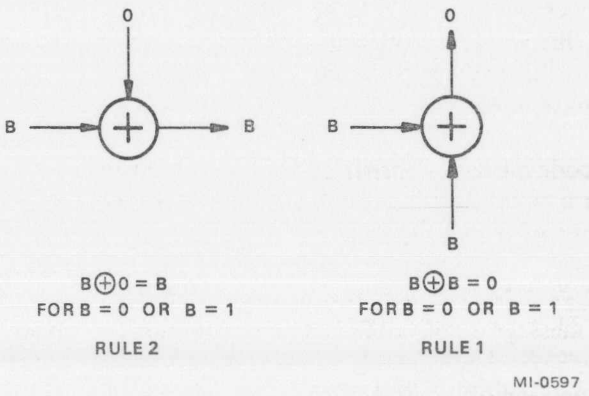


Figure 13-4. Logical Rules for Exclusive-OR Gates

As the first bit of the received CRC character is applied to point A of the receiver CRC logic, the exclusive-OR gate obeys rule 1 in Figure 13-4 and produces a 0 if the first bit of the calculated CRC matches the first bit of the CRC being received. All of the other exclusive-OR gates in the Figure 13-3 style of CRC logic will obey rule 2 in Figure 13-4 and will become essentially "transparent"; a 0 will be shifted into the left end of the shift register. As long as the calculated CRC contained in the receiver CRC shift register continues to match the CRC being received from the transmitting station, the output of the right-most exclusive-OR gate will continue to be zero, and the other exclusive-OR gates will continue to be "transparent." When all of the CRC bits have been shifted, the CRC shift register bits will all be zeros. This will be true regardless of the placement of the exclusive-OR gates and hence true for all types of CRC. An exception is that SDLC and HDLC start with a pre-set value in the shift register and end with a special non-zero result.

The most important property of CRCs is that, due to the feedback arrangements, the exact state of the shift register is dependent on a great deal of past history. Hence it is highly unlikely that a burst of errors could produce a CRC calculation that was the same as that for the data as originally transmitted before the errors occurred.

CRC requires no bit per character like parity does, but there are two check characters at the end of each block of characters, since the CRC shift register is two character lengths long. So that the reader may correlate this simplified presentation with various trade journal articles which discuss CRC polynomials in mathematical terms, a mathematical presentation follows.

A cyclic code message consists of a specific number of data bits and a BCC. Let n equal the total number of bits in the message and k equal the number of data bits; then n-k equals the number of bits in the BCC.

The code message is derived from two polynomials which are algebraic representations of two binary words, the generator polynomial P(X) and the message polynomial G(X). The generator polynomial is the type of code used (CRC-12, CRC-16, and CRC-CCITT); the message

polynomial is the string of serial data bits. The polynomials are usually represented algebraically by a string of terms in powers of X such as $X^n \ldots + X^3 + X^2 + X + X^0$ (or 1). In binary form, a 1 is placed in each position that contains a term; absence of a term is indicated by a 0. The convention followed in the following presentation is to place the least significant bit (X^0) at the right. For example, if a polynomial is given as $X^4 + X + 1$, its binary representation is 10011 (3rd and 2nd degree terms are not present).

Given a message polynomial G(X) and a generator polynomial P(X), the objective is to construct a code message polynomial F(X) that is evenly divisible by P(X). It is accomplished as follows:

1. Multiply the message G(X) by X^{n-k} where n-k is the number of bits in the BCC.
2. Divide the resulting product X^{n-k} [G(X)] by the generator polynomial P(X).
3. Disregard the quotient and add the remainder C(X) to the product to yield the code message polynomial F(X), which is represented as X^{n-k} [G(X)] + C(X).

The division is performed in binary without carries or borrows. In this case, the remainder is always one bit less than the divisor. The remainder is the BCC and the divisor is the generator polynomial; therefore, the bit length of the BCC is always one less than the number of bits in the generator polynomial.

A simple example is explained below.

1. Given:

   Message polynomial G(X) = 110011 ($X^5 + X^4 + X + X^0$)
   Generator polynomial P(X) = 11001 ($X^4 + X^3 + 1$)
   G(X) contains 6 data bits
   P(X) contains 5 bits and will yield a BCC with 4 bits; therefore, n-k = 4.

2. Multiplying the message G(X) by X^{n-k} gives:

$$X^{n-k}[G(X)] = X^4(X^5 + X^4 + X + X^0) = X^9 + X^8 + X^5 + X^4$$

   The binary equivalent of this product contains 10 bits and is 1100110000.

3. This product is divided by P(X).

```
              100001  ◄— quotient
P(X)→11001 ⟌1100110000 ◄— Xn-k [G(X)]
             11001
               10000
               11001
                1001  ◄— remainder=C(X)=BCC
```

4. The remainder C(X) is added to $X^{n-k}[G(X)]$ to give F(X) = 1100111001.

The code message polynomial is transmitted. The receiving station divides it by the same generator polynomial. If there is no error, the division will produce no remainder and it is assumed that the message is correct. A remainder indicates an error. The division is shown below.

```
              100001
P(X)→11001 ⟌1100111001 ◄— F(X)
             11001
               11001
               11001
               00000  ◄— no remainder
```

A more practical example of generating a BCC by long division is shown in Figure 13-5.

In typical data communication hardware, the BCC is computed and accumulated in a shift register. The configuration of the register is based on the CRC code to be implemented. The number of stages in

the register is equal to the degree of the generating polynomia number of exclusive-OR elements is also a function of the polync In the subsequent examples, a unique register configuration is s for each CRC code (CRC-12, CRC-16, and CRC-CCITT).

```
                                                 Quotient (To Be Discarded)
                                   11111111111111101
→11000000000000101 ⟌100000000000000000000000000000000
                    11000000000000101
  Generator          10000000000001010      Message (16 Bits)
  Polynomial         11000000000000101      Plus 16 Zeros
  CRC-16              10000000000011110
  X16 + X15 + X2 + 1  11000000000000101
                       10000000000110110
                       11000000000000101
                        10000000001100110
                        11000000000000101
  Exclusive-OR           10000000011000110
  Operation              11000000000000101
                          10000000110000110
                          11000000000000101
                           10000001100000110
                           11000000000000101
                            10000011000000110
                            11000000000000101
                             10000110000000110
                             11000000000000101
                              10001100000000110
                              11000000000000101
                               10011000000000110
                               11000000000000101
                                10110000000000110
                                11000000000000101
                                 11100000000000110
                                 11000000000000101
                                   10000000000001100
                                   11000000000000101
  Remainder is BCC ———————→         1000000000001001
                                                   ↑—MSB
```

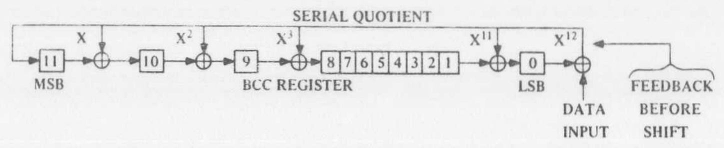
Figure 13-5. BCC Computation Using Long Division Method w CRC-16 as Generator

1. *CRC-12*
CRC-12 is applied to synchronous systems that use 6-bit characters. The BCC accumulation is 12 bits. The generator polynomial is $X^{12} + X^{11} + X^3 + X^2 + X + 1$ with prime factors of $(X+1)$ and $(X^{11} + X^2 + 1)$. It provides error detection of bursts up to 12 bits in length.

Figure 13-6 shows a block diagram configuration of a BCC register for use with CRC-12. A step-by-step shift pattern is shown as the data is serially applied to the register. Initially, the register contains all 0s. A 12-bit data word (a 1 followed by eleven 0s) is the input to the register. Prior to the first shift, the first data bit (LSB), which is a 1, is exclusive-ORed with the 0 from the LSB of the register. The result on the serial quotient line is a 1 which is sent via the feedback paths to the following places: bit 11 and the exclusive-ORs between bits 11 and 10, bits 10 and 9, bits 9 and 8, and bits 1 and 0. When the feedback settles down, the first shift takes place and produces the register states in the line labeled 1 under the SHIFT NO. column. The shift also presents the next data bit to the input. The process repeats until all 12 data bits are encoded. The BCC is the contents of the register after shift number 12. The most important fact to remember is that the exclusive-OR of the LSB of the BCC register and the input data bit set up the feedback path prior to the shifting operation. The result of this operation is shown under the column labeled FEEDBACK BEFORE SHIFT. When the shift takes place, the results of the exclusive-OR operations are shifted into the register.

The subsequent examples, which show the BCC accumulation for CRC-CCITT and CRC-16, can be analyzed the same way; the register length, data word length, and feedback paths are different but the process is the same.

2. *CRC-CCITT*
CRC-CCITT is the standard used to compute a BCC for European systems. When operating with eight-bit characters, the BCC accumulation is 16 bits. The generator polynomial is $X^{16} + X^{12} + X^5 + 1$. It provides error detection of bursts up to 16 bits in length. Additionally, more than 99% of error bursts greater than 12 bits can be detected.



SHIFT NO.							
Start	0	0	0	0 0 0 0 0 0 0 0	0	–	1
1	1	1	1	1 0 0 0 0 0 0 0	1	1	1
2	1	0	0	0 1 0 0 0 0 0 0	1	0	1
3	1	0	1	1 0 1 0 0 0 0 0	1	0	1
4	1	0	1	0 1 0 1 0 0 0 0	1	0	1
5	1	0	1	0 0 1 0 1 0 0 0	1	0	1
6	1	0	1	0 0 0 1 0 1 0 0	1	0	1
7	1	0	1	0 0 0 0 1 0 1 0	1	0	1
8	1	0	1	0 0 0 0 0 1 0 1	1	0	1
9	1	0	1	0 0 0 0 0 0 1 0	0	0	0
10	0	1	0	1 0 0 0 0 0 0 1	0	0	0
11	0	0	1	0 1 0 0 0 0 0 0	1	0	1
12	1	1	1	0 0 1 0 0 0 0 0	1	0	1

LSB

MSB

Arrows Indicate Exclusive-OR of Data Bit and LSB of BCC Register Prior to Shift

Block Check Character (BCC)

12 Bit Data Word

NOTES

□ = BCC Register Stage

⊕ = Exclusive-OR

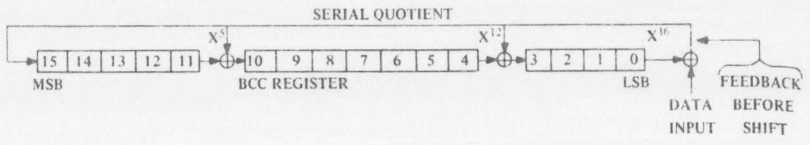
CRC-12 Polynomial = $X^{12} + X^{11} + X^3 + X^2 + X + 1$

Figure 13-6. BCC Accumulation Using CRC-12, Transmit Sequence

Figure 13-7 shows a BCC accumulation using a 16-bit data word (a 1 followed by fifteen 0s).

3. *CRC-16*
CRC-16 is applied to synchronous systems that use eight-bit characters. The BCC accumulation is 16 bits. The generator polynomial is $X^{16} + X^{15} + X^2 + 1$. It provides error detection of bursts up to 16 bits in length. Additionally, more than 99% of error bursts greater than 16 bits can be detected.

Figure 13-8 shows a BCC accumulation using a 16-bit data word (a 1 followed by fifteen 0s).



SHIFT NO.					
Start	0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0	–	1
1	1 0 0 0 0	1 0 0 0 0 0 0	1 0 0 0	1 LSB	0
2	0 1 0 0 0	0 1 0 0 0 0 0	0 1 0 0	0	0
3	0 0 1 0 0	0 0 1 0 0 0 0	0 0 1 0	0	0
4	0 0 0 1 0	0 0 0 1 0 0 0	0 0 0 1	0	1
5	1 0 0 0 1	1 0 0 0 1 0 0	1 0 0 0	0	0
6	0 1 0 0 0	1 1 0 0 0 1 0	0 1 0 0	0	0
7	0 0 1 0 0	0 1 1 0 0 0 1	0 0 1 0	0	0
8	0 0 0 1 0	0 0 1 1 0 0 0	1 0 0 1	0	1
9	1 0 0 0 1	1 0 0 1 1 0 0	1 1 0 0	0	0
10	0 1 0 0 0	1 1 0 0 1 1 0	0 1 1 0	0	0
11	0 0 1 0 0	0 1 1 0 0 1 1	0 0 1 1	0	1
12	1 0 0 1 0	1 0 1 1 0 0 1	0 0 0 1	0	1
13	1 1 0 0 1	1 1 0 1 1 0 0	0 0 0 0	0	0
14	0 1 1 0 0	1 1 1 0 1 1 0	0 0 0 0	0	0
15	0 0 1 1 0	0 1 1 1 0 1 1	0 0 0 0	0	0
16	0 0 0 1 1	0 0 1 1 1 0 1	1 0 0 0	0 MSB	0

Arrows Indicate Exclusive-OR of Data Bit and LSB of BCC Register Prior to Shift

Block Check Character (BCC)

16 Bit Data Word

NOTES

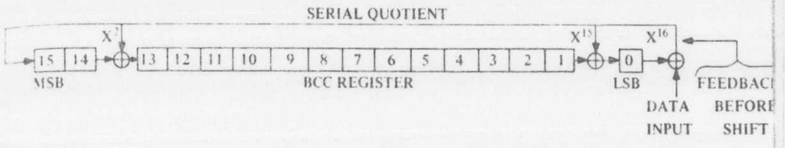
☐ = BCC Register Stage

⊕ = Exclusive-OR

CRC–CCITT Polynomial = $X^{16} + X^{12} + X^5 + 1$

Figure 13-7. BCC Accumulation Using CRC-CCITT, Transmit Sequence

The three CRC examples (Figures 13-6, 13-7, and 13-8) show BCC accumulations that are to be used in a transmission sequence. Figure 13-9 shows the BCC accumulation performed on a message that has been received along with a BCC. The message and the computed BCC have been taken from Figure 13-8, i.e., a BCC accumulation using CRC-16. In Figure 13-9, the accumulation process is the same as that shown in Figure 13-8 through shift number 16. Starting with shift number 17, the BCC is sent to the data input, LSB first. A correct transmitted BCC results in all 0s in the BCC register at the end of the computation. In effect, this is a comparison of the transmitted BCC with the one computed at the receiving station. A correct comparison yields a 0 remainder, or all 0s in the BCC register.



SHIFT NO.					
Start	0 0	0 0 0 0 0 0 0 0 0 0 0 0 0	0	–	1
1	1 0	1 0 0 0 0 0 0 0 0 0 0 0 0	1	1 LSB	1
2	1 1	1 1 0 0 0 0 0 0 0 0 0 0 0	1	0	1
3	1 1	0 1 1 0 0 0 0 0 0 0 0 0 0	1	0	1
4	1 1	0 0 1 1 0 0 0 0 0 0 0 0 0	1	0	1
5	1 1	0 0 0 1 1 0 0 0 0 0 0 0 0	1	0	1
6	1 1	0 0 0 0 1 1 0 0 0 0 0 0 0	1	0	1
7	1 1	0 0 0 0 0 1 1 0 0 0 0 0 0	1	0	1
8	1 1	0 0 0 0 0 0 1 1 0 0 0 0 0	1	0	1
9	1 1	0 0 0 0 0 0 0 1 1 0 0 0 0	1	0	1
10	1 1	0 0 0 0 0 0 0 0 1 1 0 0 0	1	0	1
11	1 1	0 0 0 0 0 0 0 0 0 1 1 0 0	1	0	1
12	1 1	0 0 0 0 0 0 0 0 0 0 1 1 0	1	0	1
13	1 1	0 0 0 0 0 0 0 0 0 0 0 1 1	1	0	1
14	1 1	0 0 0 0 0 0 0 0 0 0 0 0 1	0	0	0
15	0 1	1 0 0 0 0 0 0 0 0 0 0 0 0	1	0	1
16	1 0	0 1 0 0 0 0 0 0 0 0 0 0 0	1	0 MSB	1

Arrows Indicate Exclusive-OR of Data Bit and LSB of BCC Register Prior to Shift

Block Check Character (BCC)

16 Bit Data Word

NOTES

☐ = BCC Register Stage

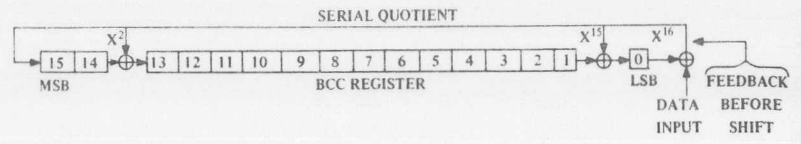
⊕ = Exclusive-OR

CRC-16 Polynomial = $X^{16} + X^{15} + X^2 + 1$

Figure 13-8. BCC Accumulation Using CRC-16, Transmit Sequence

In addition to the shift register implementations of Cyclic Redundanc Checks that are shown in the preceding discussions, it is relativel simple to implement CRC calculation using a parallel arrangement o exclusive-OR gates and two registers. The CRC accumulated to date i stored in Register 1. The character just received is stored in Register 2 A network of exclusive-OR gates combines the contents of Registers and 2 and the result is recorded back into Register 1.

Software algorithms (mainly table driven) have also been developed tc do CRC calculation.



SHIFT NO.						
Start	0 0	0 0 0 0 0 0 0 0 0 0 0 0 0	0	–		1
1	1 0	1 0 0 0 0 0 0 0 0 0 0 0 0	1	1	LSB	1
2	1 1	1 1 0 0 0 0 0 0 0 0 0 0 0	1	0		1
3	1 1	0 1 1 0 0 0 0 0 0 0 0 0 0	1	0		1
4	1 1	0 0 1 1 0 0 0 0 0 0 0 0 0	1	0		1
•	• •	• •	•	•		•
•	• •	• Same as CRC-16 Transmit •	•	•	DATA	•
•	• •	• Sequence •	•	•		•
•	• •	• •	•	•		•
13	1 1	0 0 0 0 0 0 0 0 0 0 0 1 1	1	0		1
14	1 1	0 0 0 0 0 0 0 0 0 0 0 0 1	0	0		0
15	0 1	1 0 0 0 0 0 0 0 0 0 0 0 0	1	0		1
16	1 0	0 1 0 0 0 0 0 0 0 0 0 0 0	1	0	MSB	1
17	0 1	0 0 1 0 0 0 0 0 0 0 0 0 0	0	1	LSB	0
18	0 0	1 0 0 1 0 0 0 0 0 0 0 0 0	0	0		0
19	0 0	0 1 0 0 1 0 0 0 0 0 0 0 0	0	0		0
20	0 0	0 0 1 0 0 1 0 0 0 0 0 0 0	0	0		0
21	0 0	0 0 0 1 0 0 1 0 0 0 0 0 0	0	0		0
22	0 0	0 0 0 0 1 0 0 1 0 0 0 0 0	0	0		0
23	0 0	0 0 0 0 0 1 0 0 1 0 0 0 0	0	0		0
24	0 0	0 0 0 0 0 0 1 0 0 1 0 0 0	0	0	BCC	0
25	0 0	0 0 0 0 0 0 0 1 0 0 1 0 0	0	0		0
26	0 0	0 0 0 0 0 0 0 0 1 0 0 1 0	0	0		0
27	0 0	0 0 0 0 0 0 0 0 0 1 0 0 1	0	0		0
28	0 0	0 0 0 0 0 0 0 0 0 0 1 0 0	1	0		0
29	0 0	0 0 0 0 0 0 0 0 0 0 0 1 0	0	1		0
30	0 0	0 0 0 0 0 0 0 0 0 0 0 0 1	0	0		0
31	0 0	0 0 0 0 0 0 0 0 0 0 0 0 0	1	0		0
32	0 0	0 0 0 0 0 0 0 0 0 0 0 0 0	0	1	MSB	0

Arrows Indicate Exclusive-OR of Data Bit and LSB of BCC Register Prior to Shift

BCC Register ALL 0s Received Message Assumed To Be Correct

16 Bit Data Word and BCC Received

NOTES

□ = BCC Register Stage

⊕ = Exclusive-OR.

CRC-16 Polynomial = $X^{16} + X^{15} + X^2 + 1$

Figure 13-9. BCC Accumulation Using CRC-16, Receive Sequence

CHAPTER 14 SYNCHRONOUS COMMUNICATION

In Chapter 1, the format for a typical character being transmitted in an asynchronous transmission system was discussed. It is repeated in Figure 14-1.

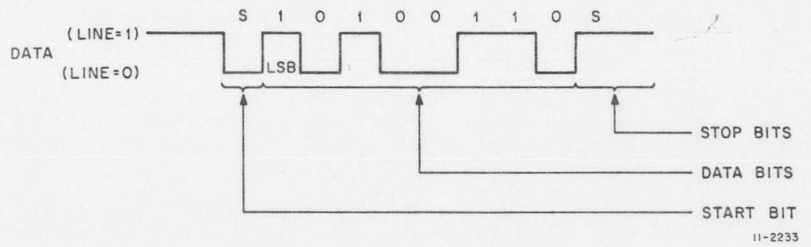


Figure 14-1. Asynchronous Data Character Format

Although modern asynchronous receivers do not require a STOP interval for mechanism coasting purposes, they do require a STOP interval to guarantee that each character will begin with a 1-to-0 transition, even if the preceding character was entirely zeros. The requirement for a 1-to-0 transition to indicate the beginning of each character causes an eight bit data character to require 10 bit times to transmit. Twenty percent of the line time is being used strictly for timing purposes.

Synchronous communication requires either a separate clock lead from the transmission point to the reception point, in addition to the